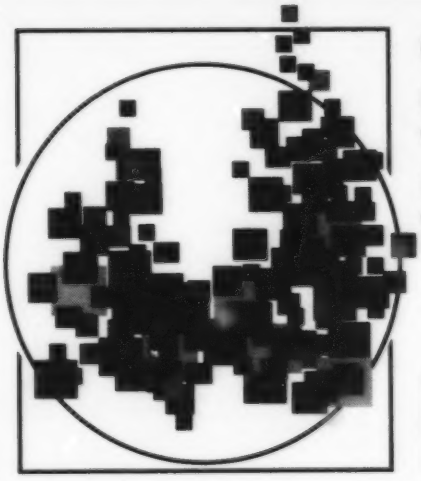

CENTRE FOR
SYSTEMS
SCIENCE

LCCR

Simon Fraser University, Burnaby, B.C., Canada V5A 1S6 / (604) 291-4704

Capture of Sufficient Bindings for Irregular Linear Recursions

Jiawei Han

CSS/LCCR TR90-22

December 1990

Capture of Sufficient Bindings for Irregular Linear Recursions

Jiawei Han
School of Computing Science
Simon Fraser University
British Columbia, Canada V5A 1S6
han@cs.sfu.ca

Abstract. *Binding propagation in recursive rules has been popularly studied in deductive database research. Some interesting techniques, such as Magic rule rewriting and its variants, have been developed based on the study of binding propagation in rule/goal graphs. In this paper, we re-examine binding propagations in linear recursions and demonstrate that the rule/goal graph may not be able to capture sufficient bindings for certain kinds of linear recursions (which we call irregular linear recursions), and hence the Magic rule rewriting technique may not be able to derive the minimal set of query-relevant facts for such recursions. We characterize such recursions using a variable connection graph and propose a compilation technique to overcome this difficulty.*

Keywords. *Databases, deductive databases, rule rewriting techniques, linear recursions, rule/goal graphs, the Magic Sets method,* and *recursive query evaluation.*

1. Introduction

Magic Sets method is one of the most important recursive query evaluation techniques [1-3, 9, 11, 13]. The method rewrites a recursion into an equivalent, but more efficiently evaluable one for a query based on the analysis of binding propagation in a rule/goal graph [1, 11]. A query is evaluated by first computing the Magic Set, the portion of the database relevant to the query, and then performing semi-naive evaluation on the derived Magic Sets.

Binding propagation in a rule/goal graph captures sufficient bindings for queries on many kinds of recursions. However, the technique can only be used to propagate bindings to future expansions *via IDB subgoals*. For some kinds of recursions, binding information should be propagated among different expansions *via EDB subgoals* as well. Such propagation cannot be caught by the rule/goal graph analysis because the graph does not register sufficient information for the propagation. As a result, a rule/goal graph-based rule rewriting technique, such as the Magic rule rewriting, is unable to

† The work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant A-3723 and a research grant from Centre for Systems Science of Simon Fraser University.

capture such kind of bindings and thus generates unnecessarily large Magic Sets for such recursions.

To overcome this difficulty, we suggest a compilation technique which is based on the expansion regularity of linear recursions [6]. The technique compiles a linear recursion into a compiled formula which captures the bindings propagated via both IDB and EDB subgoals. Efficient query evaluation plans can be generated based on the analysis of such compiled formulas.

The paper is organized as follows. In Section 2, we discuss the difficulties of regular binding propagation techniques. In Section 3, we characterize irregular linear recursions using a variable connection graph. We outline a query-independent compilation technique in Section 4 and summarize our discussion in Section 5.

2. Difficulties of Regular Binding Propagation Techniques

We assume that a **deductive database** is a database which consists of an **EDB** (a set of database relations) and an **IDB** (a set of function-free Horn-clause rules, in our discussion) [13]. A **rule** is of the form "$p_0 :- p_1, p_2, ..., p_m$." (in the syntax of Prolog), where p_i $(0 \le i \le m)$ is of the form $p_i(X_1, ..., X_n)$, in which X_j $(1 \le j \le n)$ is a variable or a constant. The head predicate p_0 is a **goal** predicate, and each predicate in the body p_i $(1 \le i \le m)$ is a **subgoal**. A rule is **recursive** if its head predicate appears also in its body (notice that we ignore indirect/mutual recursions here). It is **linearly recursive** if its head predicate appears exactly once in the body. A **linear recursion** is a program consisting of *one linear recursive rule* and one or more *nonrecursive (exit) rules*. Without loss of generality, we assume that all the nonrecursive predicates are EDB predicates.

Definitions. For a linear recursion with a recursive predicate p, we consider that the **first expansion** of p is the recursive rule itself. The **k-th expansion** of p $(k > 1)$ is the unification of the recursive rule with the $(k-1)$-th expansion of p. The **0-th expanded exit rule set** is the set of nonrecursive rules of p. The **k-th expanded exit rule set** $(k \ge 1)$ is generated by the unification of the set of nonrecursive rules of p with the k-th expansion of p. The **compiled formula** of p is the union of the set of formulas generated by all the expanded exit rules of p.

Example 1. We analyze the binding propagation of a query "$? - p(c_1, c, Y)$." on a linear recursion defined by a rule set $\{(r_0), (r_1)\}$, where c_1 and c are constants, X's and Y's are variables, and p is a recursive predicate defined by EDB predicates a, b and e.

(r_0) $p(X_1, X, Y) :- e(X_1, X, Y)$.

(r_1) $p(X_1, X, Y) :- a(X, Y), p(X_2, X_1, Y_1), b(X_2, Y_1)$.

We assume that the readers are familiar with binding propagation [11] and the Magic rule rewriting [2, 3]. According to [13], the rule/goal graph of the recursion is constructed as follows. For each goal or subgoal $p(X_1, ..., X_k)$ of arity k in a query or a rule, we create an adorned goal node p^{α}, where an *adornment*, or *binding pattern*, α, is a string of b's and f's of length k to indicate whose corresponding arguments are bound

or free. For each position in the rule, we create an adorned rule node where the adornment is a superscript of the form $[X_1, ..., X_m \mid Y_1, ..., Y_n]$ where the X's are bound and the Y's are free. For example, the adorned goal node for the query is p^{bbf}, and an adorned rule node $r_{1.0}$ is $r_{1.0}^{[X, X_1 \mid X_2, Y, Y_1]}$, where $r_{1.0}$ is the rule node of (r_1) at the 0-th position, the position in front of the first subgoal, a. The rule/goal graph for the query is shown in Fig. 1.

Following the binding propagation rules [11], the bindings in the adorned goal node, p^{bbf}, are propagated to the subgoal p in the body of the recursive rule, resulting in an adorned subgoal node, p^{fbf}, which are in turn propagated to the subgoal p in the body of the recursive rule at the next expansion, resulting in p^{fff}. Obviously, p^{fff} cannot propagate any binding information to the body of the rule. The adorned rules, (r_1'), (r_{12}') and (r_{13}'), represent such binding propagation, where (r_1') is an adorned rule of (r_1) with respect to the adorned goal node p^{bbf}, (r_{12}') is an adorned rule *unifying directly* (without renaming variables or modifying adornments) with (r_1') at the second expansion, which is called the **unifying rule** at the second expansion. Similarly, (r_{13}') is the unifying rule

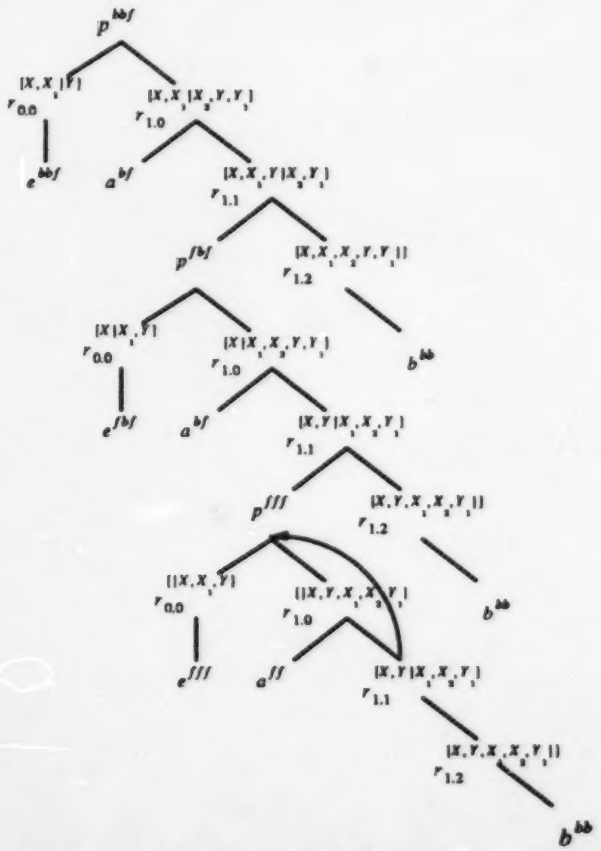

Fig. 1. The rule/goal graph for the query on the recursion of Example 1.

at the third expansion.

(r_1') $p^{bbf}(X_1, X, Y) :- a^{bf}(X, Y), p^{fbf}(X_2, X_1, Y_1), b^{bb}(X_2, Y_1).$

(r_{12}') $p^{fbf}(X_2, X_1, Y_1) :- a^{bf}(X_1, Y_1), p^{fff}(X_3, X_2, Y_2), b^{bb}(X_3, Y_2).$

(r_{13}') $p^{fff}(X_3, X_2, Y_2) :- a^{ff}(X_2, Y_2), p^{fff}(X_4, X_3, Y_3), b^{bb}(X_4, Y_3).$

Since there is no binding information which can be passed to p in the body of (r_{13}'), the Magic Set involves the entire relation A (A is the relation corresponding to the predicate a). Clearly, the binding propagation cannot reduce the set of data to be examined in the semi-naive evaluation.

We may reorder the subgoals in the body of the rule in search for better rewritten rules [12]. For example, (r_1) can be reordered into (rr_1) by switching p and b in the body.

(rr_1) $p(X_1, X, Y) :- a(X, Y), b(X_2, Y_1), p(X_2, X_1, Y_1).$

By propagation of bindings using a rule/goal graph, the corresponding adorned rule of (rr_1) is (rr_1'), and its unifying rule at the second expansion is (rr_{12}').

(rr_1') $p^{bbf}(X_1, X, Y) :- a^{bf}(X, Y), b^{ff}(X_2, Y_1), p^{bbb}(X_2, X_1, Y_1).$

(rr_{12}') $p^{bbb}(X_2, X_1, Y_1) :- a^{bb}(X_1, Y_1), b^{ff}(X_3, Y_2), p^{bbb}(X_3, X_2, Y_2).$

The reordering avoids the generation of the uninstantiated subgoal p^{fff}. However, both (rr_1') and (rr_{12}') contain an uninstantiated subgoal b^{ff}. This implies that the derivation of the Magic Set will involve the iterative processing of an uninstantiated subgoal b, i.e., the entire corresponding data relation B.

There are total 3! = 6 possible subgoal orderings in this example. Besides the above discussed two: (1) "a, p, b" (subgoal ordering), and (2) "a, b, p", we have four more choices: (3) "b, a, p", (4) "b, p, a", (5) "p, a, b" and (6) "p, b, a". It is easy to verify that all the remaining alternatives have similar binding propagation problems.

However, this does not imply that there exists no efficient binding propagation for this example. Fig. 2 shows the propagation of the binding "$X_1 = c_1$" in three rules, (r_1),

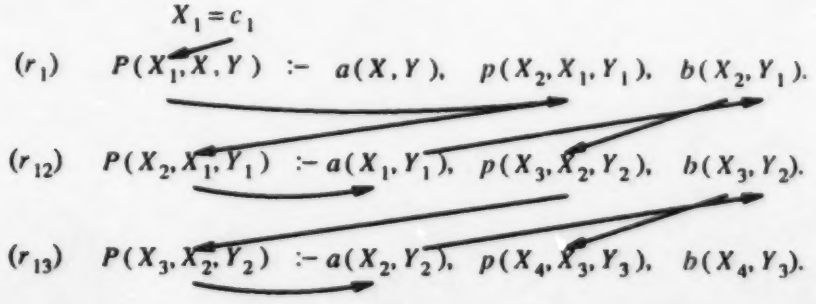

Figure 2. Propagation of binding "$X_1 = c_1$" in unifying rules.

(r_{12}), and (r_{13}), where (r_{12}) is the unifying rule of (r_1), and (r_{13}) is the unifying rule of (r_{12}). We can see that the bindings between different expansions can be propagated *via EDB subgoals*. First, query constant c_1 binds X_1, which binds Y_1 of the predicate $a(X_1, Y_1)$ in (r_{12}). The binding is propagated back to the predicate $b(X_2, Y_1)$ in (r_1), making X_2 bound. Then X_2 in the predicate $p(X_3, X_2, Y_2)$ in (r_{12}) is bound. X_2 plays the same role in (r_{12}) and (r_{13}) as X_1 in (r_1) and (r_{12}), making X_3 of the predicate $p(X_4, X_3, Y_3)$ in (r_{13}) bound. Future expansions have the similar binding propagations.

To make use of such binding propagation, we examine the second expanded rule (r_2), the rule obtained by unifying (r_1) and (r_{12}).

(r_2) $p(X_1, X, Y) :- a(X, Y), a(X_1, Y_1), p(X_3, X_2, Y_2), b(X_3, Y_2), b(X_2, Y_1).$

The subgoals of the second expanded rule (r_2) can be reordered by moving the last subgoal $b(X_2, Y_1)$ to the front of the recursive subgoal p. Then the corresponding adorned rule becomes (r_2').

(r_2') $p^{bbf}(X_1, X, Y) :- a^{bf}(X, Y), a^{bf}(X_1, Y_1), b^{fb}(X_2, Y_1), p^{fbf}(X_3, X_2, Y_2), b^{bb}(X_3, Y_2).$

Similarly, the adorned rule for the third expansion should be (r_3').

(r_3') $p^{bbf}(X_1, X, Y) :- a^{bf}(X, Y), a^{bf}(X_1, Y_1), b^{fb}(X_2, Y_1), a^{bf}(X_2, Y_2), b^{fb}(X_3, Y_2),$
$p^{fbf}(X_4, X_3, Y_3), b^{bb}(X_4, Y_3).$

Since the bindings of the recursive subgoal p in the second and the third expansions share the same adornment, p^{fbf}, future expansions will have similar binding patterns.

Let "$ab(X_1, X_2) :- a(X_1, Y_1), b(X_2, Y_1).$". The compiled formula of the recursion can be derived as (r_c), which consists of a *well-formed ab*-chain [6]. Notice that the *ab*-chain is formed by connecting two EDB subgoals a and b in two consecutive expansions.

$$(r_c)\quad p(X_1, X, Y) = e(X_1, X, Y) \cup \bigcup_{i=0}^{\infty} (a(X, Y), ab^i(X_i, X_{i+1}), e(X_{i+2}, X_{i+1}, Y_{i+1}), b(X_{i+2}, Y_{i+1})),$$

where a **well-formed chain** $ab^i(X_i, X_{i+1})$ is defined as below,

$$ab^i(X_i, X_{i+1}) = \begin{cases} true & \text{if } i = 0, \\ ab^{i-1}(X_{i-1}, X_i), ab(X_i, X_{i+1}) & \text{if } i > 0. \end{cases}$$

We then examine the evaluation of the query, "$?- p(c_1, c, Y).$". The compiled formula indicates that the query can be evaluated without using any uninstantiated subgoals since bindings can be propagated via a sequence of *ab*'s. The answer to the query consists of two sets: the first is derived from $e(c_1, c, Y)$, and the second from (r_c').

$$(r_c')\quad \bigcup_{i=0}^{\infty} (a(c, Y), X_1 = c_1, ab^i(X_i, X_{i+1}), e(X_{i+2}, X_{i+1}, Y_{i+1}), b(X_{i+2}, Y_{i+1})).$$

The formula (r_c') can be evaluated using a typical *partial* (i.e., query-relevant) transitive closure algorithm [13] by pushing the query constant "$X_1 = c_1$" into the *ab*-chain expression. An interesting alternative is to apply an *existence checking algorithm* [5] which verifies whether c_1 may derive any answer for the chain. Observe that (r_c') is the *logical and* of the expression $a(c, Y)$ and the union of a sequence of the following expressions,

(r_{e_1}) $e(X_2, c_1, Y_1), b(X_2, Y_1)$.

(r_{e_2}) $ab(c_1, X_2), e(X_3, X_2, Y_2), b(X_3, Y_2)$.

(r_{e_3}) $ab(c_1, X_2), ab(X_2, X_3), e(X_4, X_3, Y_3), b(X_4, Y_3)$.

.....

Since the above sequence does not contain any variable appearing in the query predicate, the evaluation returns *true* if *any* expression in the above sequence is evaluated to *true*, or *false* if *every* expression in the above sequence is evaluated to *false*. Therefore, the processing terminates as soon as one query expression in the above sequence is evaluated to true. That is, we first evaluate (r_{e_1}) and terminate if there is any answer in the evaluation. Otherwise, evaluate (r_{e_2}) and terminate if there is any answer in the evaluation, etc. In the worst case, it evaluates the partial transitive closure of AB, where AB is the relation for ab. Obviously, in most cases the evaluation terminates long before the partial transitive closure is completely computed. □

Example 2. We examine one more example. Suppose a query "$?- p(c, c_1, X_2)$." is posed to the recursion defined by (t_0) and (t_1), where c and c_1 are constants, X's are variables, and p is a recursive predicate defined by EDB predicates a, b, d and e.

(t_0) $p(X, X_1, X_2) :- e(X, X_1, X_2)$.

(t_1) $p(X, X_1, X_2) :- a(X_1, X_2), p(X_1, X_2, X_3), b(X), d(X_3)$.

Similar to Ex. 1, the adorned rule of (t_1) with respect to the query is (t_1').

(t_1') $p^{bbf}(X, X_1, X_2) :- a^{bf}(X_1, X_2), p^{bbf}(X_1, X_2, X_3), b^{b}(X), d^{b}(X_3)$.

Since the adornment of p is the same in the head and the body of (t_1'), the Magic Set should be the partial transitive closure of A. However, if we examine (t_{12}), the unifying rule of (t_1) at the second expansion, we will find that the bindings of X_1 can be propagated via the EDB subgoals $a(X_1, X_2)$ of (t_1), $b(X_1)$ of (t_{12}), $a(X_2, X_3)$ of (t_{12}), and $d(X_3)$ of (t_1).

(t_{12}) $p(X_1, X_2, X_3) :- a(X_2, X_3), p(X_2, X_3, X_4), b(X_1), d(X_4)$.

Let "$ab(X_1, X_2) :- a(X_1, X_2), b(X_1)$.", "$ad(X_2, X_3) :- a(X_2, X_3), d(X_3)$.", and "$abd(X_2, X_3) :- a(X_2, X_3), b(X_2), d(X_3)$.". We can easily derive the compiled formula of the recursion, (t_c), which consists of one well-formed *abd*-chain, where a well-formed chain is defined similarly as in Ex. 1.

$$(t_c)\quad p(X, X_1, X_2) = e(X, X_1, X_2) \cup (b(X), a(X_1, X_2), e(X_1, X_2, X_3), d(X_3)) \cup$$

$$\bigcup_{i=1}^{\infty} (b(X), ab(X_1, X_2), abd^{i-1}(X_i, X_{i+1}), ad(X_{i+1}, X_{i+2}), e(X_{i+1}, X_{i+2}, X_{i+3}), d(X_{i+3})).$$

The compiled formula indicates that the answers to the query "$?-p(c, c_1, X_2)$" consist of three sets: (i) $e(c, c_1, X_2)$, (ii) "$b(c), a(c_1, X_2), e(c_1, X_2, X_3), d(X_3)$", and (iii) the following sequence of query expressions.

(t_{e_1}) $b(c), ab(c_1, X_2), ad(X_2, X_3), e(X_2, X_3, X_4), d(X_4)$.

(t_{e_2}) $b(c), ab(c_1, X_2), abd(X_2, X_3), ad(X_3, X_4), e(X_3, X_4, X_5), d(X_5)$.

(t_{e_3}) $b(c), ab(c_1, X_2), abd(X_2, X_3), abd(X_3, X_4), ad(X_4, X_5), e(X_4, X_5, X_6), d(X_6)$.

.....

In the worst case, the evaluation of the above sequence needs the derivation of the partial transitive closure of ABD, which is still smaller than the partial transitive closure of A because b and d enforces constraints on the predicate a. Similar to Ex. 1, an existence checking evaluation usually terminates long before the partial transitive closure is completely computed. This is because for each value of X_2 in $ab(c_1, X_2)$, it is in the answer set if *any* of the above sequence is evaluated to true. □

3. Characterization of Irregular Recursions

The above two examples show that bindings for certain recursions should be propagated not only via IDB subgoals but also *via EDB subgoals among different expansions*. The rule/goal graph registers sufficient information for binding propagation among EDB/IDB subgoals *within* one expansion. However, it does not register the information for possible sharings of the variables in EDB subgoals *among* different expansions. Therefore, it cannot capture the bindings propagated via EDB subgoals in recursive rule expansions. Such a problem is not unique to the rule/goal graph analysis method. Interested readers may verify that many other methods [2] such as the SLD resolution [14], the OLDT resolution [4], the query/subquery approach [15], and the Alexander technique [10], have similar problems at handling these examples.

Definitions. A linear recursion in which bindings *must* be propagated *via both IDB and EDB subgoals among different expansions* is an **irregular linear recursion**, otherwise it is a **regular linear recursion**.

Most linear recursions in practical applications are regular linear recursions. Here we show that both regular and irregular linear recursions can be characterized by a variable connection graph, the *V-graph* [6], a modified α-graph of Ioannidis [8]. The V-graph was introduced in [6] in the study of compilation of linear recursions. Here we briefly introduce the graph as follows.

Definition. A **V-graph** is a hybrid, weighted and labeled graph defined as $G(V, E, W, L)$ where

1. V is a set of vertices. A vertex is associated with each variable appearing in the rule.
2. E is a set of directed and undirected edges. Direct edges (*D-edges*) connect vertices at the same argument position of the recursive predicates in the head and the body of the rule (with the arrow from the one in the head to the one in the body). Undirected edges (*U-edges*) connect consecutive vertices in the same nonrecursive predicates.
3. W is the *weight* of the edge. We associate with a U-edge weight 0 and with a D-edge $(X \rightarrow Y)$ weight 1 if traversed in the $(X \rightarrow Y)$ direction or -1 if traversed in the reverse direction.
4. L is a set of *labels* representing relations associated with vertices or edges. For each EDB predicate $p(..., X, Y, ...)$, a label p is assigned to a U-edge $(X - Y)$. For a unary EDB predicate $q(X)$, a small circle is drawn surrounding the vertex X and a label q is assigned to the circle.

Obviously, every rule has a corresponding V-graph. The V-graph of the recursive rule in a linear recursion is called *the V-graph of the recursion*.

Definitions. A **component** is a set of connected edges in the V-graph. A component is an **acyclic path** if it contains at least one D-edge but no cycle. A cycle is a **nonuniform cycle** if the D-edges in the cycle are in different directions, otherwise it is a **uniform cycle**. The **length** of a cycle (or a path) is the absolute value of the algebraic summation of the weights of all the D-edges in the cycle (or path) by traversing the cycle (or path) in one direction.

Example 3. The V-graph of the rule (r_1) consists of one nonuniform cycle of length 1 (Fig. 3 (r_1)); that of the rule (t_1) consists of two acyclic paths connecting to a uniform cycle of length 1 (Fig. 3 (t_1)); and that of the *twisted* same-generation rule (sg_1) (where p is for *parent*) [2] consists of one uniform cycle of length 2 (Fig. 3 (sg_1)). □

(sg_0) $sg(X, X)$.

(sg_1) $sg(X, Y) :- p(X, X_1), sg(Y_1, X_1), p(Y, Y_1)$.

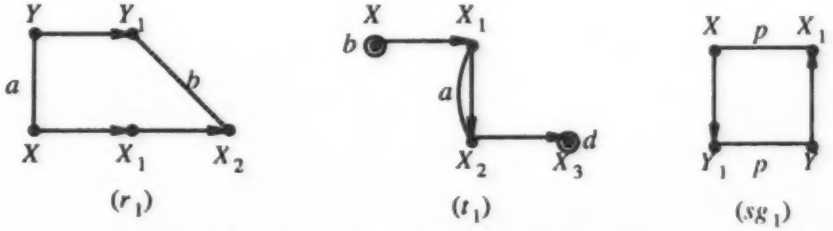

Figure 3. The V-graphs of the linear recursions.

Definition. A V-graph component is an **irregular component** if it forms (by itself) an irregular linear recursion, otherwise it is a **regular component**.

As a generalization of Ex. 1, we have,

Theorem 1. A linear recursion whose V-graph consists of a nonuniform cycle of non-zero length is an irregular linear recursion.

Proof sketch.

An example of Theorem 1 is the recursion $\{(r_0), (r_1)\}$. We prove the general case. Since a nonuniform cycle consists of D-edges in different directions, there must exist a U-edge which connects the tails of two D-edges. (Otherwise, the variables in the head predicate which correspond to the tails of the two D-edges are undefined: *unsafe*). Suppose these two D-edges are $X \rightarrow X_1$ and $Y \rightarrow Y_1$. There must exist another D-edge, such as $Z \rightarrow Z_1$ with its tail connected to either X_1 or Y_1. (Otherwise, the cycle is of zero-length, and it is a bounded recursion [8]). Suppose the D-edge is connected to Y_1. Such a V-graph is shown in Fig. 4, which corresponds to the linear recursive rule (u_1).

(u_1) $p(X, Y, Z, \cdots) :- a(X, Y), p(X_1, Y_1, Z_1, \cdots), b(Y_1, Z), g(X_1, T_1), \cdots.$

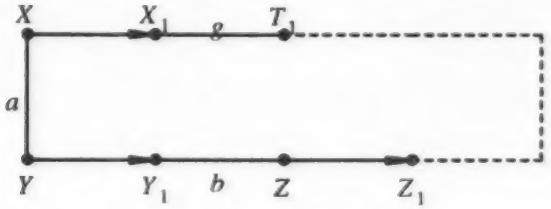

Figure 4. The V-graph of a nonuniform cycle.

Obviously, its unifying rule at the second expansion should be (u_{12}).

(u_{12}) $p(X_1, Y_1, Z_1, \cdots) :- a(X_1, Y_1), p(X_2, Y_2, Z_2, \cdots), b(Y_2, Z_1), g(X_2, T_2), \cdots.$

Suppose a query contains a binding for Z. The binding of Z can be propagated to Y_1, X_1 and T_1 between the two expansions via the EDB subgoals $b(Y_1, Z)$ of (u_1), $a(X_1, Y_1)$ of (u_{12}), and $g(X_1, T_1)$ of (u_1). Without binding propagation via EDB subgoals between the two expansions, the binding of Z cannot be propagated to T_1. Thus it is necessary to propagate the binding of Z *via both IDB and EDB subgoals* among different expansions, and the recursion is an irregular linear recursion. □

Similarly, we can prove the corollary below (a generalization of Ex. 2),

Corollary 1.1. A linear recursion whose V-graph consists of an acyclic path with one end connecting to a uniform cycle of length 1 is an irregular linear recursion.

Similarly, we have,

Corollary 1.2. A linear recursion whose V-graph consists of multiple disjoint components of which at least one is irregular is an irregular linear recursion.

Corollary 1.3. A linear recursion whose V-graph contains an acyclic path with one end connecting to an irregular component is an irregular linear recursion.

We examine what kinds of linear recursions are regular recursions.

Theorem 2. A linear recursion whose V-graph consists of a uniform cycle is a regular linear recursion.

Proof sketch.

An example of Theorem 2 is the recursion $\{(sg_0), (sg_1)\}$. We examine a linear recursive rule (v_1), whose V-graph (Fig. 5) consists of a uniform cycle.

(v_1) $p(X, Y, Z, \cdots) :- a(X_1, Y), b(Y_1, Z), ..., g(T_1, X), p(X_1, Y_1, Z_1, \cdots)$.

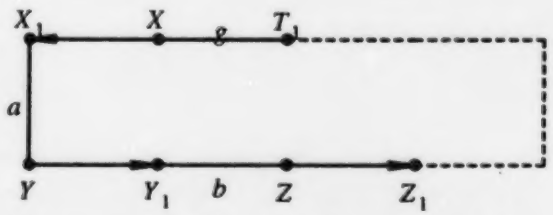

Figure 5. The V-graph of a uniform cycle.

Obviously, its unifying rule at the second expansion should be (v_{12}).

(v_{12}) $p(X_1, Y_1, Z_1, \cdots) :- a(X_2, Y_1), b(Y_2, Z_1), ..., g(T_2, X_1), p(X_2, Y_2, Z_2, \cdots)$.

Suppose a query contains a binding for Z. The binding of Z is propagated to Y_1 in $b(Y_1, Z)$ of (v_1) and then to X_2 in $a(X_2, Y_1)$ of (v_{12}) via the IDB subgoal $p(X_1, Y_1, Z_1, \cdots)$. However, X_2 cannot be propagated via other EDB subgoals in either (v_1) or (v_{12}). Bindings for other variables have the similar behavior. Thus all the bindings can be captured by propagating *via IDB subgoals only* to future expansions. Similar arguments apply to the uniform cycle with fewer U-edges. Therefore, it is a regular linear recursion. □

Similarly, we have the following corollaries.

Corollary 2.1. A linear recursion whose V-graph consists of multiple disjoint regular components is a regular linear recursion.

Corollary 2.2. A linear recursion whose V-graph consists of a set of U-edges linking *within* and/or *among* regular components is still regular.

Corollary 2.3. A linear recursion whose V-graph consists of an acyclic path linking (regular or irregular) components can be irregular.

Example 4. We present examples for the last two corollaries. Suppose that the exit rule of all the recursions here is (s_0). The recursion, $\{(s_0), (s_1)\}$, whose V-graph (Fig. 6 (s_1)) consists of a U-edge connection *within* a uniform cycle is regular. The recursion, $\{(s_0), (s_2)\}$, whose V-graph (fig. 6 (s_2)) consists of a U-edge connection *among* two uniform cycles is also regular. Their regularity can be verified by readers.

(s_0) $p(Y, Y, Z) :- e(X, Y, Z)$.

(s_1) $p(X, Y, Z) :- a(X, Z_1), p(X_1, Y_1, Z_1), b(X_1, Y, Y_1, Z)$.

(s_2) $p(X, Y, Z) :- a(X, Y_1), p(Y, Y_1, Z_1), b(Y, Z, Z_1)$.

We show that the recursion, $\{(s_0), (s_3)\}$, whose V-graph (Fig. 6 (s_3)) consists of a D-edge connecting two uniform cycles, is an irregular recursion.

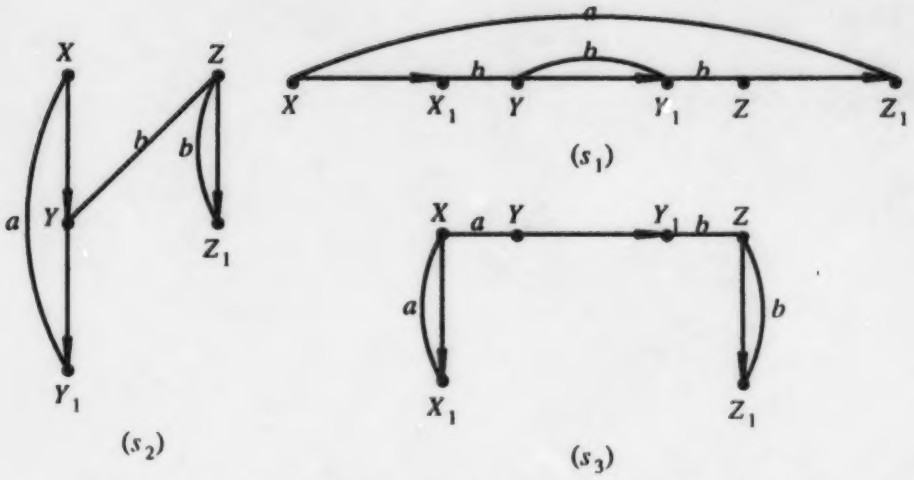

Figure 6. The V-graphs of rules (s_1), (s_2) and (s_3), respectively.

(s_3) $p(X, Y, Z) :- a(X_1, X, Y), p(X_1, Y_1, Z_1), b(Y_1, Z, Z_1).$

(s_{31}) $p(X_1, Y_1, Z_1) :- a(X_2, X_1, Y_1), p(X_2, Y_2, Z_2), b(Y_2, Z_1, Z_2).$

(s_3') $p^{bff}(X, Y, Z) :- a^{fbf}(X_1, X, Y), p^{bff}(X_1, Y_1, Z_1), b^{bfb}(Y_1, Z, Z_1).$

This can be verified by examining the binding passing of $X = c$ with an adorned goal p^{bff}. The rule/goal graph analysis derives the adorned rule (s_3'), which indicates that the Magic Set for the query should be the partial transitive closure of A. However, (s_{31}), the unifying rule of (s_3), indicates that the bindings can be further propagated via EDB subgoals between the two expansions. Since "$a(X_1, X, Y)$" in (s_3), "$a(X_2, X_1, Y_1)$" in (s_{31}) "$b(Y_1, Z, Z_1)$" in (s_3), and "$b(Y_2, Z_1, Z_2)$" in (s_{31}) make all the EDB predicates connected, the compilation results in a transitive closure formula. Thus the query can be evaluated by computing a partial transitive closure on the *instantiated* A and B relations, which avoids the semi-naive evaluation on the *uninstantiated* relation B using the Magic Sets method. Thus the recursion is an irregular linear recursion. □

Notice that the theorems and corollaries, though covering many popularly encountered V-graph configurations, do not cover all the possible cases. A detailed study of the compilation of linear recursions by the analysis of V-graphs is in [6].

4. Query-Independent Compilation of Linear Recursions

Regular linear recursions can be handled by the rule/goal graph-based compilation techniques [13]. However, irregular linear recursions require more powerful binding propagation techniques. Clearly, it is desirable to develop an integrated solution to handle both regular and irregular linear recursions uniformly.

From the examples presented in the paper, it is clear that a compilation technique should be developed which takes care of binding propagations via both IDB/EDB subgoals among different expansions. Our analysis shows that the regularity of the

expansions of linear recursions has close relationship with the configurations of the V-graphs. A compilation technique has been developed based on the classification [16] and analysis of different V-graph components [6], which shows that a linear recursion can be compiled into either an n-chain recursion ($n \geq 1$) (a potentially infinite set of relational expressions consisting of n well-formed chains) or a bounded recursion (a finte set of relational expressions). Therefore, linear recursions can be compiled into highly regular compiled formulas which represent the possible connections of EDB subgoals among different expansions. Moreover, the compilation of recursions and the generation of compiled formulas can be automated based on the simulation of the recursive rule expansion process using a variable connection matrix, *V-matrix* (a matrix with similar configurations as V-graphs). The maximum number of matrix expansions required in the simulation of recursion expansions is no greater than n, the arity of the recursive predicate. A detailed study of the automated generation of compiled formulas for linear recursions will be presented in a separate paper.

The compilation of a recursion into a set of compiled formulas independent of the possible query forms is called the *query-independent compilation* of a recursion. Query-independent compilation of linear recursions facilitates the systematic development of efficient evaluation techniques for vairous kinds of queries. This is because the compilation results in highly regular chain- or bounded- formulas, which not only captures the bindings which could be difficult to capture by other techniques but also provides precise chain connection information (in relational expression forms) for query optimization. Both qualitative and quantitative analyses can be performed on the compiled chains to determine the start end of the chain, the evaluation directions, the possible existence checking evaluation [5], the pushing of query constraints at the start end of a chain, and the possible pushing of other constraints into the chain for efficient evaluation. A detailed study of the constraint-based query evaluation of compiled linear recursions is in [7].

5. Conclusions

We studied a special kind of linear recursions, irregular linear recursions, whose binding information should be propagated via both IDB and EDB subgoals among different expansions. Regular binding propagation techniques may not be able to capture sufficient bindings for some queries on irregular linear recursions. We characterize such recursions using a variable connection graph [6] and suggest a query-independent compilation technique to compile linear recursions into compiled formulas. The compiled formulas capture the variable sharing information among different expansions and facilitate query processing on general linear recursions.

Our study is confined to linear recursions. Obviously, the *irregularity* of binding propagation discovered in linear recursions should exist in more complex kinds of recursions, such as mutual recursions, nonlinear recursions, etc. It is an interesting research topic on the compilation of such kind of recursions to capture sufficient bindings for query analysis and optimization in deductive databases.

References

1. F. Bancilhon, D. Maier, Y. Sagiv and J. D. Ullman, Magic Sets and Other Strange Ways to Implement Logic Programs, *Proc. 5th ACM Symp. Principles of Database Systems*, Cambridge, MA, March 1986, 1–15.
2. F. Bancilhon and R. Ramakrishnan, An Amateur's Introduction to Recursive Query Processing Strategies, *Proc. 1986 ACM-SIGMOD Conf. Management of Data*, Washington, DC, May 1986, 16–52.
3. C. Beeri and R. Ramakrishnan, On the Power of Magic, *Proc. 6th ACM Symp. Principles of Database Systems*, San Diego, CA, March 1987, 269–283.
4. F. Bry, Query Evaluation in Recursive Databases: Bottom-Up and Top-Down Reconciled, *Proc. 1st Int. Conf. Deductive and Object-Oriented Databases (DOOD'89)*, Kyoto, Japan, December 1989, 20–39.
5. J. Han, Multi-Way Counting Method, *Information Systems*, **14**(3), 1989, 219–229.
6. J. Han, Compiling General Linear Recursions by Variable Connection Graph Analysis, *Computational Intelligence*, **5**(1), 1989, 12–31.
7. J. Han, Constraint-Based Reasoning in Deductive Databases, *Proc. 7th Int. Conf. Data Engineering*, Kobe, Japan, April 1991.
8. Y. E. Ioannidis, A Time Bound on the Materialization of Some Recursively Defined Views, *Proc. 11th Int. Conf. Very Large Data Bases*, Stockholm, Sweden, August 1985, 219–226.
9. R. Ramakrishnan, Magic Templates: A Spellbinding Approach to Logic Programs, *Proc. Int. Conf. Logic Programming*, Seattle, WA, August 1988, 140–159.
10. H. Seki, On the Power of Alexander Templets, *Proc. 8th ACM Symp. Principles of Database Systems*, Philadelphia, PA, March 1989, 150–159.
11. J. D. Ullman, Implementation of Logical Query Languages for Databases, *ACM Trans. Database Syst.*, **10**(3), 1985, 289–321.
12. J. D. Ullman and M. Y. Vardi, The Complexity of Ordering Subgoals, *Proc. 7th ACM Symp. Principles of Database Systems*, Austin, TX, March 1988, 74–81.
13. J. D. Ullman, *Principles of Database and Knowledge-Base Systems, Vol. 2*, Computer Science Press, Rockville, MD, 1989.
14. L. Vieille, A Database-Complete Proof Procedure Based on SLD-Resolution, *Proc. 4th Int. Conf. Logic Programming*, Melbourne, Australia, 1987, 70–103.
15. L. Vieille, From QSQ towards QoSaQ: Global Optimization of Recursive Queries, *Proc. 2nd Int. Conf. Expert Database Systems*, Vienna, VA, April 1988, 743–778.
16. C. Youn, L. J. Henschen and J. Han, Classification of Recursive Formulas in Deductive Databases, *Proc. 1988 ACM-SIGMOD Conf. Management of Data*, Chicago, IL, June 1988, 320–328.

Centre for Systems Science

• Simon Fraser University established the **Centre for Systems Science** as a provincial centre for advanced study. The objective of CSS is twofold: coordinate and combine advanced systems research at SFU, and encourage application of research results within industry in order to further the technological development of British Columbia.

•CSS has assembled three research groups to streamline its efforts. The orientation of these groups is as follows: intelligent systems, microelectronics, and computer and communications systems. Because of shared research interests, this latter group merges with LCCR (**Laboratory for Computer and Communications Research**) for the publication of technical reports.

• CSS's three groups involve over 80 skilled faculty researchers principally from SFU's Schools of Engineering Science and Computing Science, but also draw substantially on Mathematics, Physics, Geography, Philosophy, Linguistics, Psychology, Communication, and Kinesiology.